キャパシティ分析

キャパシティ分析

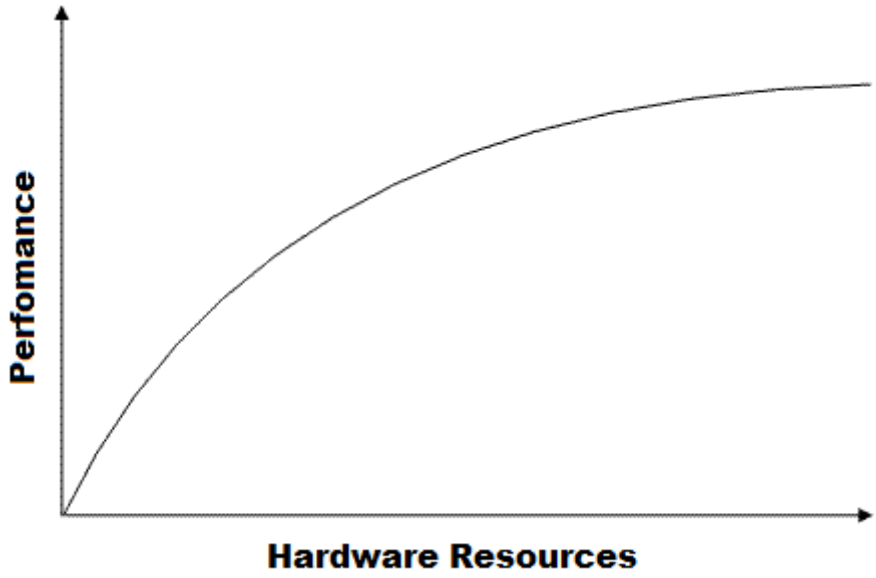
Pandora FMS ドキュメント一覧に戻る

キャパシティ分析

概要

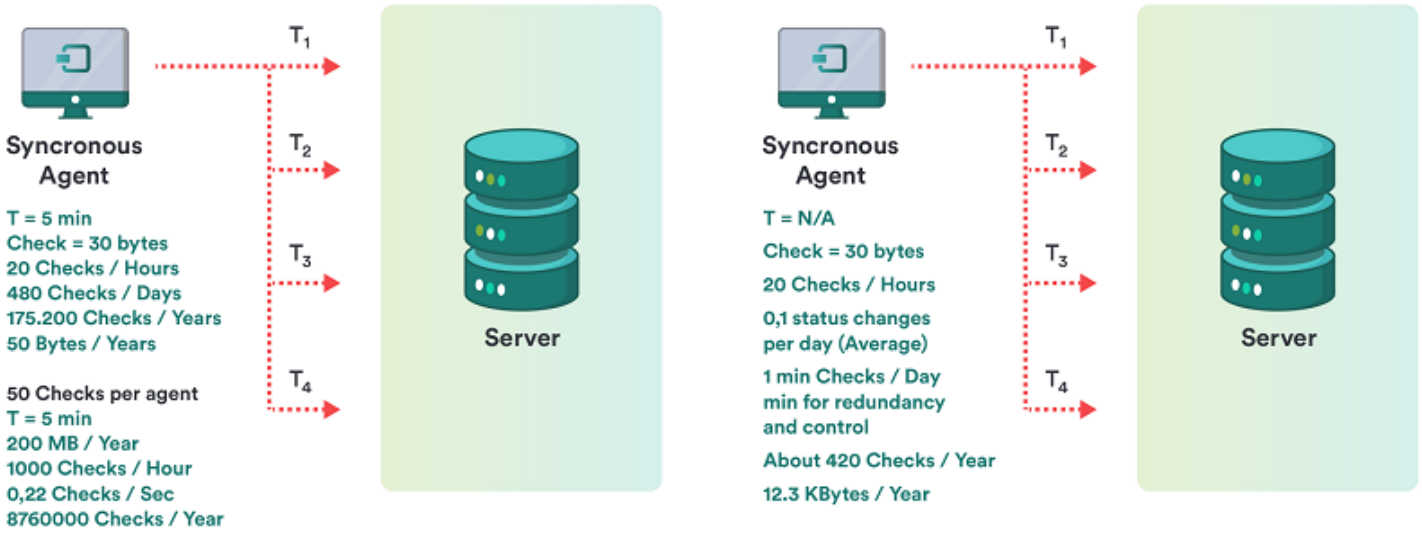
Pandora FMS は、正しく設定しないとボトルネックになりうる複数の要素がある複雑なアプリケーションです。ここでは、特定のキャパシティを得るための必要条件を知るために、特定のパラメータに関する Pandora FMS のスケーラビリティの詳細を見ていきます。

最初は、クラスタを使ったシステムを対象にしたロードテストです。データベースクラスタの中に一つの Pandora サーバがあります。ロードテストは、サーバ一台ごとの最大のキャパシティを見るのに便利です。現在(バージョン 3.0以降)のアーキテクチャではN台の個別のサーバと一つのメタコンソールでスケーラビリティはリニアに伸びますが、一台のサーバではそうはなりません。(次のグラフに示します。)



データストレージと圧縮

実際に Pandora は、リアルタイムでデータを圧縮しています。これは、保存されるデータ量を計算するためにはとても重要です。最初に、データの保存方法に関して従来のシステムと Pandora FMS の“ 非同期 ”のデータ保存方法の違いについてみてみます。以下に図を示します。



従来のシステム

チェックを一日平均20実施すると、1年で 5MB の容量が必要です。エージェントあたり 50チェックでは、1年 250MB になります。

従来とは異なる Pandora FMS のような非同期システム

チェックにおいて 1日平均 0.1の状態変化では、1年で 12.3KBの容量になります。エージェントあたり 50チェックでは、1年で 615KB です。

用語の定義

次に、より理解を深められるように、ここで使われている用語について説明します。

- 情報の断片化: Pandora FMS が扱う情報によりパフォーマンスは変化します。情報には、常に変化するもの(CPUの使用率など)や、固定(あるサービスの状態など)のものがあります。Pandora FMS は、これらをデータベースに圧縮して保存します。これはパフォーマンスおよびキャパシティに対して重要な要素となります。そこで、より断片化し、データベースにたくさん保存し、処理量を増やすことが、同じ情報を処理するために必要となります。
- モジュール: モニタリングにおいて情報を収集する基本的な単位です。他では、イベントであったり、監視項目、メトリックとも言われるものです。
- 間隔: 一つのモジュールで情報を収集する時間間隔です。通常、分または秒で表されます。“平均”値は通常 5分です。
- アラート: データがしきい値を越えたり状態が 障害 や 警告 になったときに、Pandora FMS が実行する通知です。

容量考察の例

これまでの考察と対象範囲

次の 3種類のパターンで設定を行う場合を考えてみます。

- ステージ1: 500エージェントでの設定
- ステージ2: 3000エージェントでの設定
- ステージ3: 6000エージェントでの設定

このデータ量において、正しく Pandora FMS の要求スペックを決めるためには、どのような種類のモニタリングをする予定であるかを知る必要があります。次の例では“QUASAR TECNOLOGIES” という架空の会社の特徴を示しています。

- 90% のモニタリングをソフトウエアエージェントで実施。
- 技術/ポリシーでグループ化できる似たようなシステムがある。
- モニタするモジュールやイベント間で、実行間隔が異なる。
- 大量の非同期情報がある(イベントやログなど)。
- あまり変化がない状態を確認する処理がたくさんある。
- 全体的にパフォーマンスに関する情報は少ない。

すべての技術的な内容とその実装に関する調査(システムとそのモニタ方法の確認)の結果、次のような結論に至ります。

- 1システムあたり、平均して 40個のモジュールやイベントが存在する。
- 平均モニタリング間隔は、1200秒(20分)である。
- 5分ごとに情報を送ってくるモジュールもあれば、1週間に一度だけのモジュールもある。
- 全グループの全モジュール (240,000) のうち、確認のたびに変更が発生する可能性があるのは 25% である。
- モジュールごとのアラート比率は 1.3 (モジュール/イベントごとに 1.3 アラート) である。
- アラートの発生率は、1% と仮定する。(これは我々の経験上の予測です)

この結論は、予測を策定するための基本となります。理解しやすいように Excel シートにまとめてみます。

Probability of change (%)	**25**	**Probability Trigger Alert**	**1.00%**
Average interval (sec)	**1200**		
Modules per agent	**40**		
Alerts / Module	**130.00%**		
Total Alerts	*52*		

これらの初期データに対して必要な計算をあてはめます。データベースのサイズおよび、モニタリングに必要となる一秒間あたりのモジュール実行数、その他パラメータを予測できます。

Milestone	Agents	Total Modules	Rate mod./sec.	Alerts	Modules/day	MB/Month	Modules/month	Pack./day
Phase 1	500	20.000	16.67	26,000	360,000	1,030	10,800,000	144,000
	1,000	40.000	33.33	52,000	720,000	2,060	21,600,000	288,000
	2,000	80,000	66.67	104,000	1,440,000	4,120	43,200,000	576,000
Phase 2	3,000	120,000	100	156,000	2,160,000	6,180	64,800,000	864,000
	4,000	160,000	133.33	208,000	2,880,000	8,240	86,400,000	1,152,000
	5,000	200,000	166.67	260,000	3,600,000	10,300	108,000,000	1,440,000
Phase 3	6,000	240,000	200	312,000	4,320,000	12,360	129,600,000	1,728,000
	10,000	400,000	333.33	520,000	7,200,000	20,599	216,000,000	2,880,000

キャパシティの考察

それぞれのフェーズで実装のための基本的な要求事項 (秒あたりのモジュール実行数)、アラートの数、日ごとのモジュール実行数、月ごとの容量がわかったら、本番に近いシステムで実際にサーバの負荷テストを行います。(ここでのテストは、本番に近いシステムでは実行できていません。)

これらの負荷テストでは、Pandora FMS の処理能力がわかり、どの程度で性能劣化するかがわかります。これは、次のような目的において便利です。

1. 対象のハードウエアで最大どれくらいの規模まで対応できるかを推定する場合。
2. ストレージの限界および、ヒストリ DB へ情報を移すポイントを知りたい場合。
3. サービス停止や計画停止により、処理する情報がたまった場合の最大処理量に対する余裕を知りたい場合。
4. モニタ対象の情報の変化率が変わった場合のパフォーマンスに与える影響を知りたい場合。
5. 大量のアラート処理の影響を知りたい場合。

テストは、Intel Core Duo プロセッサ 2.5GHz および、メモリ 2GB を積んだ DELL のサーバ PowerEdge T100 にて実施しました。このサーバでは、Ubuntu Server 8.04 が動いており、HA 環境のテストに使っています。テストは、QUASAR TECHNOLOGIES プロジェクトに似たエージェント設定で実施しました。同じハードウエアではありませんが、同じような HA 環境で、WUASAR TECHNOLOGIES と似たパフォーマンスの影響および大量のデータを扱うことによるその他問題(主に利便性)を評価できる環境です。

Agents	6005	Interval	3000	Modules per agent	30
Total modules	180150	Change prob.%	100	Alerts per agent	20

	Probability of change (100%)			Probability of change (25%)		
	Rate Pack./sec.	Rate mod./sec.	Rate processed	Rate Pack./sec.	Rate mod./sec.	Rate processed
Day 1	3	90	149,88%	8	240	399,67%
Day 2	2,85	85,5	142,38%	7,5	225	374,69%
Day 3	2,71	81,23	135,26%	6,75	202,5	337,22%
Day 4	2,57	77,16	128,50%	6,08	182,25	303,50%
Day 5	2,44	73,31	122,07%	5,47	164,03	273,15%
Day 6	2,32	69,64	115,97%	4,92	147,62	245,83%
Day 7	2,21	66,16	110,17%	4,43	132,86	221,25%
Day 8	2,1	62,85	104,66%	3,99	119,57	199,12%
Day 9	1,99	59,71	99,43%	3,59	107,62	179,21%
Day 10	1,89	56,72	94,46%	3,23	96,86	161,29%
Day 11	1,8	53,89	89,74%	2,91	87,17	145,16%
Day 12	1,71	51,19	85,25%	2,62	78,45	130,65%
Day 13	1,62	48,63	80,99%	2,35	70,61	117,58%
Day 14	1,54	46,2	76,94%	2,12	63,55	105,82%
Day 15	1,46	43,89	73,09%	1,91	57,19	95,24%
Day 16	1,39	41,7	69,44%	1,72	51,47	85,72%
Day 17	1,32	39,61	65,96%	1,54	46,33	77,14%
Day 18	1,25	37,63	62,67%	1,39	41,69	69,43%
Day 19	1,19	35,75	59,53%	1,25	37,52	62,49%
Day 20	1,13	33,96	56,56%	1,13	33,77	56,24%
Day 21	1,08	32,26	53,73%	1,01	30,39	50,61%

得られた結果はとても良く、システムは非常に過負荷にもかかわらず、非常に興味深い情報量を処理することができました(180,000モジュール、6000エージェント、120,000アラート）。これから得られた結論は次の通りです。

1. リアルタイムの情報は、最大15日間でヒストリーデータベースへ移動させるべきである。一週間より古いデータを動かすことがベストです。これにより、より早い動作が保障されます。

2. 情報量を考慮して、処理の余裕は想定能力よりも高い最大能力の 50% ほどである。

3 システムを構築する場合のパフォーマンスと必要な容量を決定するには、データの細分化の割合がとても重要である。

詳細方法論

前の章は、“ データサーバ ”モジュールのみに基づく“ 迅速な ”調査でした。このセクションでは Pandora FMS 容量の分析を行うためのより完全な方法を示します。

出発点として、すべての選択の場面において、最悪のシナリオを想定しています。 それができない場合、それは“ 一般的な考察 ”になります。 “ 最良の場合 ”は考慮していません

次のステップは、監視のタイプまたは情報の出所に基づいて、システム容量を計算する方法です。

データサーバ

前のポイントで見たように、特定の目的に基づいて考えます。合計 3000 のエージェントに分散された 100,000 モジュールの負荷、つまり平均でエージェントあたり 33モジュールでどのような負荷で動作するかを確認したいと想定します。

pandora_xmlstress のタスクを作成し、cron または手動スクリプトを介して実行します。33個のモジュールがあり、次のような設定で展開されます。

- 文字列タイプの 1モジュール
- generic_proc タイプの 17モジュール
- generic_data タイプの 15モジュール

generic_proc タイプの 17個のモジュールのしきい値を次のように設定します。

```
module_begin
module_name Process Status X
module_type generic_proc
module_description Status of my super-important daemon / service / process
module_exec type=RANDOM;variation=1;min=0;max=100
module_end
```

generic_data タイプの 15個のモジュールでは、しきい値を定義する必要があります。 手順は次の通りです。

このタイプのデータが生成されるように、generic_data タイプの 15個のモジュールのしきい値を設定する必要があります。

```
module_exec type=SCATTER;prob=20;avg=10;min=0;max=100
```

これらの 15個のモジュールのしきい値を設定すると、次のパターンになります。

```
 0-50 normal
 50-74 warning
 75- critical
```

pandora_xml_stress の設定ファイルにいくつかの新しいトークンを追加して XML 生成からのしきい値を定義できるようにします。 重要: Pandora FMS は、モジュールの作成時にしきい値の定義を採用するだけで、新しいデータの更新には利用されません。

```
module_min_critical 75
module_min_warning 50
```

pandora_xml_stress を実行します。

中断することなく少なくとも 48時間実行し、次のパラメーターを(Pandora FMS エージェントを使用して)監視する必要があります。

キューに入っているデータ数:

```
find /var/spool/pandora/data_in | wc -l
```

Pandora FMS サーバ CPU 使用率:

```
ps aux | grep "/usr/bin/pandora_server" | grep -v grep | awk '{print $3}'
```

pandora_server トータルメモリ:

```
 ps aux | grep "/usr/bin/pandora_server" | grep -v grep | awk '{print $4}'
```

mysqld による CPU 使用率(実行の構文を確認してください□MySQL ディストリビューションによって異なります。)

```
ps aux | grep "sbin/mysqld" | grep -v grep | awk '{print $3}'
```

Pandora FMS データベース平均応答時間:

```
/usr/share/pandora_server/util/pandora_database_check.pl
/etc/pandora/pandora_server.conf
```

不明状態の監視項目数:

```
echo "select SUM(unknown_count) FROM tagente;" | mysql -u pandora -p<password> -
D pandora | tail -1
```

(<password> は pandora ユーザのパスワードです。)

最初の実行は、サーバと MySQL 設定を“ 調整 ”するのに役立つはずです。

スクリプト /usr/share/pandora_server/util/pandora_count.sh を使用して、データの処理速度をカウントします(XML ファイルの処理が保留されている場合)。 目的は、生成されたすべてのデータ(3000)を、制限時間(5分)の 80% 未満の間隔で処理できるようにすることです。これは、3000個のデータを 4分で処理する必要があることを意味します。

```
3000 / (4x60) = 12.5
```

Pandora FMS がこの情報を処理できることを合理的に確認するには、最低 12.5 個のデータ処理速度が必要です。

調整要素:

- スレッド数
- 中間キュー内のアイテムの最大数(max_queue_files)
- もちろん、適用可能な MySQL のすべてのパラメーター(非常に重要)

これの重要性：強力なマシンに“ デフォルト ”でインストールされた GNU/Linux サーバ 1台で、Pandora は、毎秒 5〜6 データを超えることはできませんが、十分に“ 最適化 ”および“ 調整 ”された強力なマシンでは、毎秒 30-40 データまで処理することができます。また、各エージェントに含まれるモジュールの数にも依存します

/usr/share/pandora_server/util/pandora_db.pl にあるデータベースメンテナンススクリプトが毎日ではなく 1時間ごとに実行されるように、システムを設定します。

```
mv /etc/cron.daily/pandora_db /etc/cron.hourly
```

データジェネレーターを最低 48時間動かして、システムを動作させたままにします。 この時間が経過したら、次の点を評価する必要があります。

1. システムは安定しているか?、ダウンしていないか? 問題がある場合は、取得したメトリック(主にメモリ)のログとグラフを確認してください。
2. メトリック“ 不明な状態の監視項目数 ”の時間の傾向を評価します。 何らかの傾向やピークもあってはいけません。それがある場合は問題が発生しているはずです。問題が1時間の規則性で発生する場合、それはデータベース管理プロセスの並行処理に問題があるためです。
3. メトリック“ Pandora データベースの平均応答時間 ”を評価します。 時間とともに増加することはありませんが、一定のままである必要があります
4. メトリック“pandora_server CPU使用率 ”を評価します。多くのピークがあるはずですが、一定の傾向があり、上昇していないことです。
5. メトリック“MYSQL サーバ CPU使用率 ”を評価します。 多くのピークがありますが、一定の傾向で、上昇していないことです。

アラートの影響の評価

すべてが問題なければ、アラート実行パフォーマンスの影響を評価します。

障害(CRITICAL) 状態の場合、各エージェントの 5つの特定のモジュール(generic_data タイプ)に 1つのアラートを適用します。イベントの作成や syslogへの書き込みなど、それほど重要ではないもの(高い遅延の影響を回避するため、待ち時間が長いもの、たとえば電子メールメッセージを送信するようなものは避ける)を利用します。

オプションで、1つのイベント相関アラートを作成して、これら 5つのモジュールのいずれかを使用するエージェントの障害状態に対して 1つのアラートを生成します。

これらの基準の下でシステムを 12時間稼働させ、前の基準に従って影響を評価します。

データ削除と移動の評価

データストレージのポリシーが以下の通りであると仮定します。

- 72時間以上経過した古いイベントを削除
- 7日以上経過したデータをヒストリデータベースへ移動

長期的なパフォーマンスを評価するために、システムを少なくとも 10日間動作させる必要があります。 データがヒストリデータベースに移動される 7日後に“ ピーク ”が見られます。 このパフォーマンス低下は、重要な考慮点です。確認のための時間がそれほど多くとれない場合は、イベント削除を 2日に、データのヒストリデータベースへの移動を 2日に変更して、この影響を評価できます(“ 実環境 ”とは少しことなりますが)。

ICMP サーバ(Enterprise)

これは、ICMPネットワークサーバ用です。オープンソースのネットワークサーバのテストを行う場合は、ネットワークサーバ(汎用)の対応する章を参照してください。

サーバがすでに設定され動作していると仮定すると、そのパフォーマンスに関するいくつかの重要なパラメーターは次のとおりです。

```
block_size X
```

これは、システムが 1回に実行する “ping” の数を定義します。 多くの ping を同時に実行する場合は、数をかなり多く、つまり 50 から 70 に増やすことができます。

逆に、ping モジュールが少なく、対象のネットワークが大きく異なり遅延時間もそれぞれ異なるような場合は、テストに遅い方の時間を要するため、大きな数値は設定しない方が良いです。15 から 20 などの非常に低い数を使用します。

```
icmp_threads X
```

明らかに、スレッドが多いほど、実行できるチェックも多くなります Pandora FMS が実行するすべてのスレッドを追加しても、30～40 を超えることはありません。利用している GNU/Linux バージョンとハードウェアの種類に大きく依存しますが、ここでは 10 を超えるスレッドは使用するべきではありませ

次に、テストする架空の数の ping モジュールを“ 作成 ”します ping タイプのモジュール合計 3000個をテストするとします。 これを行うための最良のオプションは、すべての ping をサポートするネットワーク内のシステムを選択することです(GNU/Linux サーバならどれでも大丈夫です)。

Pandora CSV インポータ(Enterprise 版で利用可能)を使用して、次の形式でファイルを作成します。

```
(Agent name, IP,os_id,Interval,Group_id)
```

以下のシェルスクリプトを使用して、ファイルを生成できます(宛先 IP とグループ ID を変更します)。

```
A=3000
while [ $A -gt 0 ]
do
    echo "AGENT_$A,192.168.50.1,1,300,10"
    A=`expr $A - 1`
done
```

このすべての処理を開始する前に、Pandora FMS サーバが実行および監視され、前述のメトリック(CPU使用率(pandora および mysql)、不明状態のモジュール数、およびその他の興味深い監視項目)を測定する必要があります。

CSV をインポートして 3000エージェントを作成します。数分かかります。 その後、最初のエージェント(AGENT_3000)に移動し、PING タイプのモジュールを作成します。

一括操作ツールに移動し、そのモジュールを他の 2999 エージェントにコピーします。

その後、Pandora FMS はそれらのモジュールの処理を開始します。 前のケースと同じメトリックを測定し、それがどのように推移するかを評価します。 目的は、必要な ICMP タイプのモジュールの数に対して、不明状態にならずに処理できかです。

SNMP サーバ(Enterprise)

これは SNMP Enterprise ネットワークサーバーに関するものです。 オープンソースのネットワークサーバのテストの場合は、(汎用の)ネットワークサーバの章を参照してください。

サーバがすでに設定され動作していると仮定して、サーバが機能するためのいくつかの重要なパラメーターについて説明します。

```
block_size X
```

これは、システムが 1回に実行する SNMP リクエストの数を定義します。サーバが宛先 IP でグループ化することを考慮するため、このブロックは単なる指標です。 大きくしないことをお勧めします(最大 30～40)。ブロック内の要素に障害が発生すると、内部カウンターは Enterprise サーバでそれを再試行し、試行回数が x 回を超えても応答がない場合は、オープンソースのサーバに処理を渡します。

```
snmp_threads X
```

使用するハードウェアの種類と GNU/Linux のバージョンによって異なりますが、10を超えるスレッドを使用しないでください。

テストのよりすばやく実施する方法は、SNMP デバイスを使用して、すべてのインターフェイス、す

べての一連の“基本”監視モジュールを適用することです。これはSNMPエクスプローラアプリケーション(SNMP Explorer)を介して行います。 インターフェイスを特定し、すべてのメトリックを各インターフェイスに適用します。24ポートスイッチでは、これにより650モジュールが生成されます。

他の名前で同じ IP を持つ他のエージェントを作る場合は、さらに 650 モジュールを追加できます。すべてのモジュールを、同じ IP を持つすべてのエージェントにコピーすることにより、コピーされたモジュールは同じスイッチにアクセスします。

他のオプションとしては、たとえば Jalasoft SNMP Device Simulator のような SNMP エミュレータを使用することです。

ここでの目的は、SNMP モジュールプールを少なくとも 48時間一定の方法で監視し、インフラストラクチャを監視して、1秒あたりのモジュール数の監視率が一定であり、 サーバが不明状態のモジュールを生成しないことです。不明状態は、次の原因で発生する可能性があります。

- リソース(メモリ、CPU)の不足。これらのメトリクスの傾向が継続的に上昇していることが確認できた場合、それは悪いシグナルです。
- 時々発生する問題: 日次サーバの再起動(ログローテーション用)、データベースの定期保守処理の実行、またはサーバまたはデータベースサーバで実行されるその他のスクリプト。
- 監視処理とは関係ない(ネットワーク内のサーバのバックアップなどの)ネットワークの問題で、ネットワークの速度に影響を与えている場合。

プラグイン、ネットワーク(オープンソース)、HTTP サーバ

前述と同じ概念を適用しますが、より単純化された方法です。 以下を確認する必要があります。

- スレッド数
- タイムアウト(最悪の場合の発生率を計算するため)
- 平均時間の確認

これらのデータを使用してテストグループをスケーリングし、サーバ容量が時間の経過とともに一定であることを確認します。

トラップ受信

このケースはより単純です。システムが一定量のトラップを受信するのではなく、一時的に大量のトラップが来た場合の応答を評価し、そこからアラートを生成することを想定します。

それには、制御された方法で高速でトラップを生成する単純なスクリプトを実行するだけで済みます。

```
#!/bin/bash
TARGET=192.168.1.1
while [ 1 ]
do
```

```
   snmptrap -v 1 -c public $TARGET .1.3.6.1.4.1.2789.2005 192.168.5.2 6 666
1233433 .1.3.6.1.4.1.2789.2005.1 s "$RANDOM"
done
```

注: 数百のトラップがすばやく生成されるため、数秒後に CTRL+C キーで停止してください。

環境をセットアップしたら、次のことを検証する必要があります。

1. 一定速度のトラップを受信します(前述のスクリプトの while ループ内に sleep 1 を設定するだけで、1トラップ/秒を生成します)。システムを 48時間稼働させ、サーバへの影響を評価します。
2. 大量トラップ。大量のトラップ受信が発生した場合の、その前後、発生中の評価をします。
3. 巨大なトラップテーブル(5万以上)に対するシステム影響の確認。 これには、データベースメンテナンスへの影響も含みます。

イベント

SNMP の場合と同様に、次の 2つの場合の Pandora FMS システムのイベントを評価します。

1. イベント受信の通常の範囲。 これはデータサーバですでにテストされているため、ステータスが変更されるたびにイベントが生成されます。

2. 大量のイベント生成。これを行うには CLI を介してイベントの生成を強制します。 次のコマンドを使用します(作成された “TestingGroup” を使用)。

```
/usr/share/pandora_server/util/pandora_manage.pl \
  /etc/pandora/pandora_server.conf --create_event "Event test" system
TestingGroup
```

このコマンドは、トラップの生成に使用したものと同様にループを使用し、1秒ごとに数十のイベントを生成するために使用します。発生数を増やすために、複数のインスタンスを使用して 1つのスクリプトを並列化することができます。 これは、大量のイベントが発生した場合にシステムのパフォーマンスをシミュレートするのに役立ちます。 このようにして、大量イベントの前後、最中にシステムをチェックできます。

ユーザの同時アクセス

こにには WEB 監視機能を使用して Pandora FMS から独立した別のサーバを使用します。 次のタスクをこの順序で実行するユーザセッションを実行し、それらにかかる時間を確認します。

1. コンソールへのログイン。
2. インベントの参照。
3. グループ表示へ行く。
4. エージェント詳細表示へ行く。
5. サポートの参照(HTML にて)。このレポートには、レポートタイプが合計または平均のグラフのペアとモジュールのペアが含まれている必要があります。各項目の間隔は、1週間または 5日である必要があります。

6. 組み合わせグラフの参照(24時間)。
7. PDF でのレポート生成(他のレポートにて)。

このテストは、少なくとも 3人の異なるユーザを使用して行います。このタスクは並列化して毎分実行することができるため、5つのタスク(それぞれがユーザを含む)があるのであれば、5人のユーザの同時ナビゲーションをシミュレートします。環境をセットアップしたら、次のことを考慮する必要があります。

1. 各モジュールの平均速度で、メンテナンススクリプトの実行など、他の平行して行われる処理に関連した“ボトルネック”を特定します。
2. CPU とメモリの影響は、同時セッションごとにサーバで測定します。
3. 残りのセッションの平均時間を確認してシミュレートされた各ユーザセッションの影響を測定します。つまり、同時の追加セッションごとに何秒の遅延が追加されるかを見積もる必要があります。

Pandora FMS ドキュメント一覧に戻る